VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Vizualizace učení a výstupů neuronové sítě

Visualisations for Artifical Neural Network Outputs and Training

2019 Martin Výlet

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Martin Výlet**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2601R013 Telekomunikační technika

Téma: Vizualizace učení a výstupů neuronové sítě
Visualisations for Artifical Neural Network Outputs and Training.

Jazyk vypracování: čeština

Zásady pro vypracování:

Neuronové sítě se stávají nedílnou součástí softwaru pro širokou škálu použití. K nalezení optimální neuronové sítě a jejímu naučení je vhodné vizualizovat řadu statistik. Cílem práce je vytvoření webového vizualizačního rozhraní pro neuronové sítě. Nástroj umožní interaktivní zobrazení struktury neuronové sítě, průběh učení neuronové sítě, vývoj přesnosti klasifikace i charakteristik již naučené sítě. Součástí bude také implementace vizualizace vstupních datových sad neuronové sítě.

Body zadání:
1. Studium nástrojů a frameworků pro vizualizaci dat.
2. Implementace vizualizací na základě dat z neuronové sítě
3. Otestování nástroje na existujících průběžích učení neuronových sítí pro rozpoznání SIP útoků.
4. Dokumentace vizualizačního nástroje.

Seznam doporučené odborné literatury:

[1]KIRK, Andy. Data visualisation: a handbook for data driven design. Sage, 2016.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jakub Šafařík, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. 4. 2019 .....

Rád bych na tomto místě poděkoval Ing. Jakubovi Šafaříkovi, Ph.D. za cenné rady připomínky při zpracování této práce.
Poděkování také patří mé rodině a Bohu, kteří mě po celou dobu studia podporovali.

Abstrakt

Cílem této bakalářské práce je vytvoření aplikace pro vizualizaci dat z umělé neuronové sítě ve webovém prostředí pro možnou budoucí integraci s nástrojem pro práci s touto umělou neuronovou sítí, dále otestování nástroje na datech z existující umělé neuronové sítě. Proto byly pro aplikaci zvoleny jazyky HTML a JavaScript a knihovny pro práci s grafikou a grafy pro jazyk JavaScript. Aplikace umožňuje vizualizaci topologie samotné umělé neuronové sítě, zobrazení grafů vývoje ceny a přesnosti učení umělé neuronové sítě, zobrazení Matice záměn a analýzu vstupních a výstupních dat.

Klíčová slova: vizualizace, umělá neuronová síť, javascript, bakalářská práce

Abstract

The purpose of this bachelor thesis is to create application for visualisation of data of artificial neural network in web environment for possible future integration with tool for management of this artificial neural network, test this application with data from existing artificial neural network. Because of these circumstance it was decided to use HTML and JavaScript languages and libraries for graphics and graphs in JavaScript language. Application can visualise topology of the artificial neural network itself, display graphs of progress of precision and accuracy of learning of artificial neural network, display Confusion matrix and analyse input and output data.

Key Words: visualisation, artificial neural network, javascript, bachelor thesis

Obsah

Seznam použitých zkratek a symbolů

CSV	–	Comma-separated values
JSON	–	JavaScript Object Notation
PNG	–	Portable Network Graphics
JPG	–	Joint Photographic Experts Group
SVG	–	Scalable Vector Graphics
HTML	–	Hyper Text Markup Language

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Neuronové sítě se v poslední době stávají čím dál běžnější součástí softwaru. Strojové učení pomocí neuronových sítí nám umožňuje zpracovat obrovské množství dat, rozpoznávat řeč, obraz, modelovat chování mas, poruchovost vozidel MHD a mnoho dalšího.

V této práci se budu zabývat vytvořením nástroje pro vizualizace vstupů, výstupů, učení a vnitřní struktury neuronové sítě pro rozpoznávání SIP útoků. Ucelený nástroj pro tuto síť, která funguje v rámci projektu na Katedře telekomunikační techniky Fakulty elektrotechniky a informatiky, dosud neexistuje. Nástroj bude využíván v rámci tohoto projektu, je možná následná integrace do webového rozhraní pro práci s umělou neuronovou sítí.

V teoretické části rozebírám samotný pojem vizualizace, vlastnosti a zásady pro tvorbu vizualizací, dále pak základní vlastnosti jazyka JavaScript, jeho historií a použitím, zejména pro tvorbu grafiky a grafů. Poslední oddíl teoretické části se zabývá Maticí záměn, základního nástroje pro analýzu klasifikace umělou neuronovou sítí.

Aplikace měla být vyvinuta a otestována nad reálnými daty z umělé neuronové sítě.

Výsledná aplikace je rozdělena do několika částí. V první části se zabývám zobrazením vnitřní struktury umělé neuronové sítě, rozložením neuronů a hodnot jejich vazeb.

V druhé části aplikace se zabývám zobrazením průběhu učení – vývojem ceny a přesnosti neuronové sítě.

Třetí část aplikace se zabývá výpočtem a zobrazením Matice záměn a výpočty statistických informací k reálným výsledkům klasifikace neuronovou sítí vůči kontrolní sadě.

Čtvrtá část se zabývá zobrazením dat z výstupních tříd neuronové sítě – nespojový XY bodový graf s možností použití logaritmického měřítka os grafu.

Pátá část aplikace se zabývá zobrazením vstupních tříd neuronové sítě – stejný graf jako ve čtvrté části. Dále je zde zobrazen poměr zastoupení jednotlivých tříd ve vstupním souboru v koláčovém grafu a statistické informace k položkám jednotlivých tříd.

Poslední část aplikace se zabývá zobrazením tabulky s buňkami obarvenými podle binárních hodnot ve vstupním souboru.

Dále bylo mým úkolem vytvořit programátorskou dokumentaci k aplikaci.

2 Teorie

2.1 Vizualizace

Vizualizace je dnes široce používaný pojem. Spence odkazuje na slovníkovou definici pojmu vizualizovat – zformovat mentální model nebo mentální obraz něčeho.[3] Vizuální reprezentace mají dlouhou a úctyhodnou historii při výměně informací a faktů. Ale jen zhruba 20 let uplynulo od doby, kdy se vizualizace stala samostatnou nezávislou vědní disciplínou.[1] V roce 1987 byl pojem vizualizace představen pány McCormickem a kol. Definovali vizualizaci následovně:

> „Vizualizace je metoda informatiky. Transformuje obrazné do geometrického, což umožňuje výzkumníkům pozorovat jejich simulace a výpočty. Vizualizace nabízí metody jak vidět neviděné. Obohacuje proces vědeckých objevů, podporuje hluboké a nečekané vhledy do problémů."[2]

Cílem této nové vědní disciplíny je integrovat nevídané možnosti lidského vidění a obrovský výpočetní výkon počítačů a podpořit uživatele v analýze, porozumění a komunikaci s jejich daty, modely a koncepty. Aby toho bylo dosaženo, musí být splněna 3 hlavní kritéria:

- výmluvnost
- efektivita
- vhodnost

Výmluvnost odkazuje na požadavek zobrazení přesně těch informací, která obsahují data, nic víc a nic míň nesmí být vizualizováno. Efektivita primárně zvažuje úroveň, do kterého vizualizace adresuje rozpoznávací schopnosti lidského vidění, ale také zvažuje aplikační pozadí a další kontextové informace pro zobrazení intuitivně rozeznatelné a interpretovatelné vizuální reprezentace. Konečně, vhodnost zahrnuje poměr cena-výkon pro posouzení výhod vizualizačního procesu versus dosažení daného cíle. Zatímco hodnota vizuální reprezentace se neurčuje snadno (zdroj), cena je často vztažená k časové efektivitě (např. procesorový čas) a prostorové efektivitě (např. rozlišení výsledného obrazu). Tyto tři výše uvedená jsou kritéria, která by každá vizualizace měla splňovat. Proto vizualizační proces nade všechno musí počítat s dvěma aspekty: data a zadaný úkol. Jinými slovy musíme odpovědět na 2 otázky: "Co musí být prezentováno?"a "Proč to musí být prezentováno?".[1]

2.1.1 Co?

Existují různé přístupy k charakterizování vizualizovaných dat. Podle pánů Wonga a Bergerona, kteří ve svém článku „30 let multidimenzionálních multivariačních vizualizací" přišli s pojmem

„multidimenzionální multivariační data". Toto vede k rozlišení závislých a nezávislých proměnných. Nezávislé proměnné definují n-rozměrnou oblast. V této oblasti jsou pak hodnoty k závislých proměnných změřeny, spočítány nebo simulovány – definují k-variační datovou sadu. Pokud je alespoň 1 dimenze svázána s časem, říkáme tomu „časově orientovaná data". [5]

2.1.2 Proč?

Podobně jako Co? je potřeba vědět, proč jsou data vizualizována a jaký úkol se tím uživatel snaží splnit. Velmi abstraktně lze rozlišit 3 základní cíle:

- explorativní analýza
- potvrzovací analýza
- prezentování výsledků analýzy

Explorativní analýza se může zdát jako neřízené vyhledávání. K analyzovaným datům není předem dána hypotéza. Cílem je získat pochopení pro data, začít extrahovat relevantní informace a zformulovat hypotézu. Potvrzovací analýza je používána pro potvrzení nebo vyvrácení hypotézy. Můžeme tedy říct, že se jedná o řízené vyhledávání. Cílem prezentování výsledků je sdělit a šířit tyto výsledky. [4]

2.2 JavaScript

JavaScript (často zkracovaný na JS) je objektově orientovaný, interpretovaný programovací jazyk, který nefunguje sám o sobě. Je nadesignovaný ke spolupráci s jazykem HTML. Spolu tvoří interaktivní webové stránky. Dříve byly v JavaScriptu psány skoro jen a pouze klientské aplikace, v poslední době se však JavaScript dostává i na servery, např. projekt Node.JS.

2.2.1 Historie

První verze jazyka JavaScript byla vyvinuta ve společnosti Netscape v roce 1995 a to jako "otevřený multiplatformní objektový skriptovací jazyk určený k vytváření a úpravám aplikací v nezávislých sítích a na internetu". Zároveň 28 velkých technologických firem schválilo JavaScript jako otevřený multiplatformní objektový skriptovací jazyk a mělo v úmyslu poskytnout podporu tohoto jazyka ve svých produktech. [6]

Zároveň s vydáním jazyka pro klientskou stranu byl vyvíjen JavaScript i pro skriptování na serveru a později, v prosinci 1995, byl vydán JavaScript pro servery.

V roce 1996 byla zahájena standardizace u organizace European Computer Manufacturers Association (ECMA), aby ostatní prohlížeče mohly implementovat tento jazyk. Výsledkem byla oficiální specifikace jazyka ECMAScript, jehož nejznámější implementací byl právě JavaScript, dále pak ActionScrip a JScript. Následovaly standardy ECMAScript 2 a ECMAScript 3.

V roce 2005 se objevil pojem Ajax, jenž byl postaven na jazyku JavaScript a umožňoval vytvářet aplikace, kde se data načítají na pozadí bez nutnosti znovunačtení stránky a vytvářejí se tak mnohem dynamičtější aplikace. Vzniká mnoho open-source knihoven s tímto chováním, za všechny jeden příklad: jQuery.

V roce 2009 byl vydán standard ECMAScript 5, v roce 2011 standard ECMAScript 5.1 a konečně v roce 2015 ECMAScript 2015. Současně je aktuální verze ECMAScript 2018 z června 2018.[7]

2.2.2 Grafika a grafy v jazyce JavaScript

JavaScript sám o sobě neposkytuje žádné funkce pro kreslení a manipulaci s objekty grafického charakteru. Existuje ale mnoho knihoven, které tyto funkce obsahují.

Aplikace používá 2 různé typy těchto knihoven:

- grafické knihovny
- knihovny kreslící grafy

Pro vývoj aplikace byla použita knihovna p5.js. Tato knihovna obsahuje základní metody pro práci s plátnem, nastavením jeho barvy, pozice, velikosti atp. Dále obsahuje objekty představující základní geometrické obrazce, např. čára, kruh, trojúhelník, čtverec. Také obsahuje metody pro nastavení štětce, jeho barvy, tloušťky, metody pro práci s oknem a ošetření jeho změn. Příklad: změna okna vyvolá událost, která spustí metodu windowResized() ve které má programátor možnost upravit parametry své aplikace, třeba velikost plátna. Knihovna p5.js dále obsahuje metody pro základní operace s objekty – posun, rotaci podle jedné z os x, y, z, zkosení a změnu velikosti objektu. V neposlední řadě obsahuje metody obsluhy klávesnice a myši, metody pro čtení a zápis souborů, matematické operace v grafice a práci se světly.[8] Obecně se dá říci, že grafické knihovny poskytují základní tvary, jejich transformace, obsluhu událostí a základní matematické operace v grafice. Co s tímto základem vytvoří programátor je věc druhá. Grafických knihoven existuje mnoho, většinou volně dostupných na internetu. Několik příkladů viz níže:

- p5.js
- d3.js
- Raphael
- Paper.js
- Processing.js

Knihovny kreslící grafy se specializují právě na kreslení grafů, od klasických XY bodových grafů, přes koláčové, plošné, sloupcové až po jejich kombinace. Programátor se nestará o kreslení

jednotlivých částí grafu, pouze o nastavení jednotlivých částí, např. rozsah nebo popisky os, a o správný formát vstupních dat, kterému daná knihovna rozumí. Pro vývoj aplikace byla použita knihovna Highcharts, která umožňuje modifikovat téměř všechny parametry grafu za běhu programu. Grafy vytvořené pomocí této knihovny je možno designovat pomocí kaskádových stylů (CSS). U jednoduchých grafů stačí použít nastavení automaticky určené knihovnou, u složitějších je třeba úprav, aby byl graf zřejmý na první pohled, popřípadě zobrazoval další informace čistě mimo vstupní data.[9]

Několik příkladů knihoven kreslících grafy:

- Highcharts
- Plot.ly
- ECharts
- vis.js

2.3 Umělé neuronové sítě

Umělé neuronové sítě jsou počítačové modely volně inspirovány svými biologickými protějšky, tedy skutečnými neurony v lidském mozku[13], jak je příroda stvořila. Neuronová síť je frameworkem (aplikačním rámcem) použitelným v celé škále různých algoritmů strojového učení. Takovýto systém se "učí"na základě příkladů, bez toho, aby byl systém sám o sobě naprogramován pro určitý úkol.[14] Umělá neuronová síť je složena z navzájem spojených uzlů, tzv. umělých neuronů. Každý spoj může přenášet signál z jednoho neuronu do dalšího, který tento signál zpracuje a dále signalizuje připojeným neuronům.[13] Umělá neuronová síť má obyčejně vrstvy, takové, že neurony ovlivňují pouze neurony v následující vrstvě, tj. neovlivňují neurony ve stejné vrstvě ani neurony v předchozích vrstvách. Příklad takovéto umělé neuronové sítě je na obr. 2, kde je síť se vstupní vrstvou, jednou skrytou vrstvou a výstupní vrstvou.

Přínosem aplikace vyvinuté v této práci je zobrazení mnoha vizualizací nad daty z neuronové sítě v jednom nástroji. Uživatel tedy nemusí používat více různých nástrojů nebo ručně vytvářet grafy v programech typu Microsoft Excel, stačí mu prostě nahrát data do této aplikace.

2.3.1 Zobrazení sítě

Zobrazení topologie umělé neuronové sítě s hodnotami vazeb mezi neurony umožňuje získat rychlý přehled o tom, které neurony se ovlivňují, a tedy jaký vliv má vstupní datová sada na výslednou klasifikaci. Platí zde pravidla uvedená v kapitole 2.1, tedy že vizualizace by měla být jasná na první pohled.

2.3.2 Zobrazení XY grafů

Pomocí spojových XY grafů můžeme zobrazit vývoj ceny a přesnosti umělé neuronové sítě v průběhu učení. Můžeme detekovat změny v těchto parametrech a popřípadě upravit učící sadu tak, aby se neuronová síť učila co nejefektivněji a její klasifikace byly co nejpřesnější.

2.3.3 Matice záměn

Matice záměn je běžný nástroj pro posuzování přesnosti prostorových dat [11]. Jedná se o statistický nástroj pro analýzu spárovaných pozorování. Obsah matice je sada hodnot, které odpovídají stupni shody mezi spárovanými pozorováními mezi referenční datovou sadou a naměřenou datovou sadou, v našem případě naměřená datová sada odpovídá klasifikacím neuronovou sítí. Z tohoto vyplývá, že Matice záměn je čtvercová matice $M \times M$, kde M odpovídá počtu tříd braných v úvahu. Na hlavní diagonále matice jsou umístěny korektně klasifikované prvky, zatímco mimo hlavní diagonálu jsou umístěny prvky obsahující počet záměn, tedy prvky chybně klasifikovaných tříd.[12]

S Maticí záměn také souvisí pojmy[10]

- True positive (TP) - případy, kdy se klasifikace neuronovou sítí shoduje s se skutečnou/-kontrolní třídou
- True negative (TN) - případy, kdy se klasifikace neuronovou sítí shoduje se skutečností tak, že v obou případech je klasifikace "ne tato třída", tj. $TN = \sum X - TP - FP - FN$, kde X jsou jednotlivé vzájemné klasifikace.
- False positive (FP) - případy, kdy neuronová síť klasifikovala skutečnou třídu X chybně jako třídu Y
- False negative (FN) - případy, kdy neuronová síť chybně klasifikovala třídu X jako skutečnou třídu Y

Pomocí těchto pojmů se dají spočítat další statistické informace[10]

- Četnost True positive (TP rate), též zvaná pravděpodobnost správné detekce, vyjadřuje pravděpodobnost korektní klasifikace neuronovou sítí.

$$TPrate = \frac{TP}{TP + FN}$$

- Četnost False positive (FP rate) vyjadřuje pravděpodobnost chybné detekce takové, že neuronová síť klasifikovala skutečnou třídu X chybně jako třídu Y

$$FPrate = \frac{FP}{FP + TN}$$

- Četnost False negative (FN rate) vyjadřuje pravděpodobnost chybné detekce takové, že neuronová síť klasifikovala třídu X jako skutečnou třídu Y

$$FNrate = \frac{FN}{FN + TP}$$

- Přesnost (Precision) vyjadřuje poměr korektních pozitivních klasifikací vůči všem pozitivním klasifikacím, tj. kolik pozitivních klasifikací je relevantních

$$Precision = \frac{TP}{TP + FP}$$

- Citlivost (Recall) vyjadřuje poměr korektních pozitivních klasifikací vůči všem korektním klasifikacím

$$Recall = \frac{TP}{TP + FN}$$

- F1 score (Sørensen–Dice koeficient) vyjadřuje přesnost testu

$$F1score = \frac{2TP}{2TP + FP + FN}$$

- Specifičnost (Specificity) vyjadřuje poměr korektních negativních klasifikací vůči všem negativním klasifikacím

$$Specificity = \frac{TN}{TN + FP}$$

- Geometrický průměr (G-mean) udává střední hodnotu mezi citlivostí a přesností

$$G - mean = \sqrt{Precision \times Recall}$$

- Matthewův korelační koeficient (MCC) vyjadřuje kvalitu binární klasifikace, bere v úvahu všechna data, vrací hodnoty $-1 < x < 1$, tak, že 1 vyjadřuje perfektní klasifikaci a -1 absolutní neshodu mezi skutečností a klasifikacemi neuronovou sítí

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

2.3.4 Analýza vstupních a výstupních dat

Nespojové XY grafy používané při analýze vstupních a výstupních dat neuronové sítě umožňují zobrazit vztahy mezi jednotlivými parametry datové sady. Z grafů lze velmi jednoduše určit tzv. odlehlé hodnoty, které mohou zkreslit výsledky a vést k nepřesnostem v klasifikacích neuronové sítě.

3 Aplikace

Pro aplikaci byl zvolen formát webové aplikace (HTML + JavaScript) hlavně z důvodu možné budoucí integrace s webovým nástrojem pro práci s umělou neuronovou sítí. Vizualizační nástroj zobrazuje data o umělé neuronové síti v 6 částech:

- Záložka "Network" umožňuje zobrazit danou síť tak, jak je fyzicky navržená její topologie, tj. jak jsou spojené jednotlivé neurony mezi vrstvami.
- Záložka "XY Graphs" umožňuje zobrazit grafy učení - ceny a přesnosti v průběhu učení neuronové sítě.
- Záložka "Confusion Matrix" umožňuje zobrazit Matici záměn tříd predikcí neuronové sítě vůči reálným třídám.
- Záložka "Output Class" zobrazuje 2D graf s třídami útoků na základě daných vstupů.
- Záložka "Input Class" zobrazuje podobný 2D graf s třídami útoků na základě daných vstupů, navíc zobrazuje koláčový graf s procentuálním zastoupením tříd ve vstupním souboru a statistické informace.
- Záložka "Binary Input Class" zobrazuje tabulku s buňkami obarvenými podle hodnot ve vstupním souboru.

Vizualizační nástroj nepracuje přímo s neuronovou sítí, ale se soubory, které popisují její vnitřní strukturu (soubor .json) a její výstupy (typicky soubory .csv).

3.1 Network

Aplikace umožňuje mít rychlý přehled o topologii neuronové sítě a hodnotách vazeb mezi jednotlivými neurony (pro malé sítě, tj. do 20 neuronů na vrstvu, více viz styl "BigNetwork"). Pro tuto část aplikace byl použit framework p5.js.

3.1.1 Implementace

3.1.1.1 Objekt Node

Objekt typu Node představuje 1 neuron zobrazované neuronové sítě. Zná svoji pozici, všechny objekty typu Line, kterými je spojen s okolními neurony. Dále si pamatuje, jestli na něj uživatel v daný okamžik najel myší, což slouží ke zvýraznění připojených objektů typu Line.

```
1    class Node {
2        constructor(x, y) {
3            this.x = x; //xPos
4            this.y = y; //yPos
5            this.connectedLines = []; //all lines that are connected to this
                 node
6            this.over = false; //mouse over
7        }
8        ...
9    }
```

Výpis 1: Objekt typu Node

3.1.1.2 Objekt Line

Objekt typu Line představuje spojení mezi dvěma neurony zobrazované neuronové sítě. Zná svoji aktuální barvu, objekty typu Node na svých koncích a svoji hodnotu. Objekty typu Line předpokládají, že jestliže uživatel najel myší na alespoň jeden objekt typu Node na koncích objektu typu Line, najel i na ně, a mají se tedy zvýraznit.

```
1    class Line {
2        constructor(from, to, weight) {
3            this.c; //color
4            this.from = from; //Node from
5            this.to = to; //Node to
6            this.weight = weight; //weight of the bond between nodes
7            this.over = false; //mouse over
8        }
9        ...
10   }
```

Výpis 2: Objekt typu Line

3.1.1.3 Vytvoření neuronové sítě v paměti programu

Aplikace načte data ze vstupního souboru specifického formátu, viz kapitola 3.1.2. Po načtení jsou data převedena do objektu jazyka Javascript. Následně jsou vytvořeny objekty typu Node představující neurony. Data pro počet vrstev neuronů a počet neuronů v jednotlivých vrstvách jsou specifikována v objektu vstupního souboru "sizes". Neurony jsou poté "spojeny" objekty typu Line. Data pro hodnoty jednotlivých spojů jsou rovněž specifikována v objektu vstupního souboru "weights". Všechny objekty typu Node jsou uloženy do pole nodes, objekty typu Line do

pole lines, což velmi zjednodušuje vykreslovací algoritmus. Při vytváření sítě je také nastavena velikost plátna, na které se bude kreslit, a to podle vrstvy s největším počtem neuronů ve vertikálním směru a podle počtu vrstev ve směru horizontálním. Zjistí-li aplikace, že ve vstupním souboru je specifikována tzv. Big Network, automaticky načte odpovídající styl, viz kapitola 3.1.3. Po vytvoření sítě v paměti programu jsou spočítána statistická data pro jednotlivé vrstvy (minimální a maximální hodnota vazby ve vrstvě a průměr hodnot vazeb ve vrstvě), která jsou následně zobrazena pod graf s neuronovou sítí.

3.1.1.4 Vykreslení neuronové sítě

Aplikace neustále překresluje plátno s neuronovou sítí (mimo styl Big Network, viz kapitola 3.1.3). Kreslí se od nejvzdálenějšího objektu po nejbližší.

1. Při každém překreslení se začíná s prázdným plátnem
2. Nakreslí se pozadí dle specifikovaného stylu.
3. Nakreslí se objekty typu Line obsažené v poli lines, tak že na každém objektu typu Line se zavolá metoda display(). Objekt typu Line zná své konce, barvu dle specifikovaného stylu i váhu (která ovlivňuje šířku výsledné čáry) a vykreslí se.
4. Nakreslí se objekty typu Node obsažené v poli nodes, tak že na každém objektu typu Node se zavolá metoda display(). Objekt typu Node zná svou pozici, barvu dle specifikovaného stylu a vykreslí se.
5. Nakreslí se popisky objektů typu Line, pokud to specifikovaný styl dovoluje, a to tak, že na objekty typu Line se zavolá metoda displayText().
6. Nakonec se vykreslí názvy jednotlivých vrstev umělé neuronové sítě uložené v poli columnTitles.

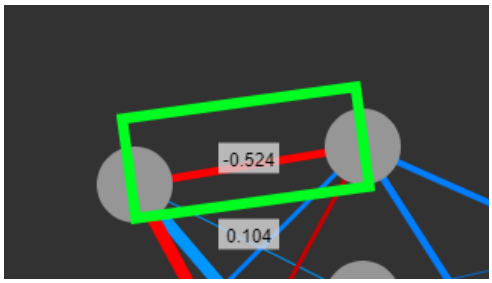
```
function draw() {
    background(backgroundColor);
    //drawing of all lines must be first because circles of nodes are above
    //we also invoke check of mouse position, to show additional info on hover
    for (let i = 0; i < lines.length; i++) {
        if (!bigNetwork)
            lines[i].rollover(mouseX, mouseY);
        lines[i].display();
    }
    //same for nodes
    for (let i = 0; i < nodes.length; i++)
        for (let j = 0; j < nodes[i].length; j++) {
            if (!bigNetwork)
                nodes[i][j].rollover(mouseX, mouseY);
            nodes[i][j].display();
        }
    //and text is above all
    if (!bigNetwork)
        for (let i = 0; i < lines.length; i++) {
            lines[i].displayText();
        }
    //columns labels
    for (let i = 0; i < columnTitles.length; i++) {
        columnTitles[i].displayText();
    }
}
```

Výpis 3: Vykreslení neuronové sítě

3.1.2 Vstupní data

Aplikace pro zobrazení topologie neuronové sítě čte data z uživatelem poskytnutého vstupního souboru. Po vybrání souboru je automaticky zahájeno nahrávání souboru a jeho následné zpracování. Vstupní soubor musí mít přesně definovanou vnitřní strukturu, jinak aplikace nebude standardně fungovat. Vstupní soubor musí být textový, s vnitřní strukturou definovanou takto: Soubor obsahuje jeden standardní JavaScriptový objekt (začíná "{" a končí "}"), který obsahuje objekt "DATA", zbylé objekty jsou ignorovány. Objekt "DATA" dále musí obsahovat objekt "sizes", který obsahuje pole s velikostmi jednotlivých vrstev, a objekt "weights", který obsahuje pole s poli s daty o hodnotách jednotlivých vazeb mezi neurony tak, že po načtení souboru

bude weights[0][0] obsahovat hodnotu vazby mezi nultým neuronem ve vstupní vrstvě a nultým neuronem v následující vrstvě. Příklad viz zelený obdélník na obr. 1.



Obrázek 1: Vazba weights[0][0]

Ve výpisu č. 4 je příklad správně strukturovaných vstupních dat.

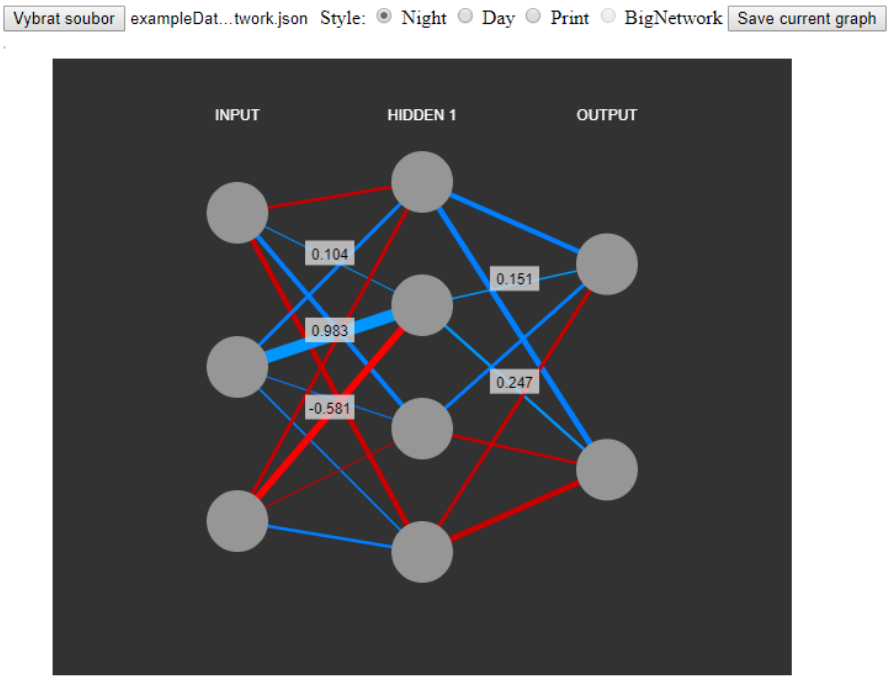
```
{
  "CLASSNAME": "neuralNetwork.ANNConfiguration",
  "DATA": {
    "sizes": [ 3, 4, 2 ],
    "weights": [
      [
        [ -0.5236576687602668, 0.590494123242314, -0.5343858108463877 ],
        [ 0.10353441863509771, 0.9826960906625022, -0.5807228036618508 ],
        [ 0.8140879945743638, 0.2560959993852605, -0.22180890788431618 ],
        [ -0.8593515456279817, 0.3644581310609274, 0.46256316578368617 ]
      ],
      [
        [ 0.8392236132244688, 0.15060505074502406, 0.5916866830811849,
          -0.6074963963312283 ],
        [ 0.9571575414523119, 0.24701436089852047, -0.4728958254531268,
          -0.957777263043057 ]
      ]
    ]
  }
}
```

Výpis 4: Správně strukturovaná vstupní data

3.1.3 Zobrazované styly

Záložka "Network" zobrazuje, jak je umělá neuronová síť navržena, jak jsou neurony propojeny a při malých umělých neuronových sítích s největší vrstvou do 20 neuronů při najetí myši, tzv.

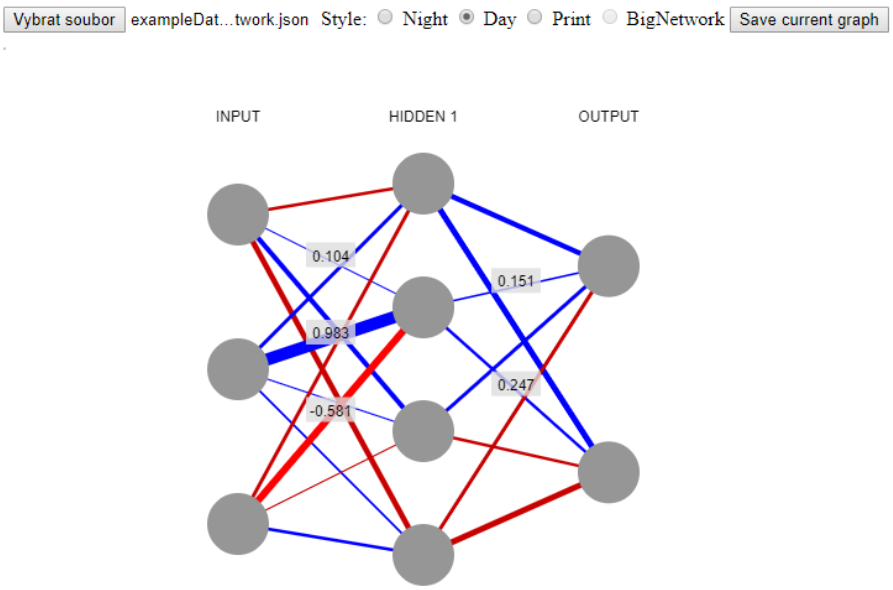
“on hover” zobrazuje hodnoty vazeb na neurony s daným neuronem spojené, dále také zvýrazní tyto vazby, příklad viz obr. 2. Tloušťka čar znázorňuje hodnoty vazeb, čím tlustší čára, tím víc se neurony ovlivňují, ať už pozitivně nebo negativně. Při zvýrazňování vazeb mezi neurony je šířka čar zdvojnásobena.



Obrázek 2: Příklad použití stylu "Night"

Uživatel má na výběr ze 3 barevných stylů zobrazení sítě:

- “Night” má tmavé pozadí, světlé barvy čar znázorňující vazby mezi jednotlivými neurony, kde světle modrá (rgb: #007EFF) znázorňuje kladnou vazbu mezi neurony, světle červená (rgb: #C80000) znázorňuje zápornou vazbu mezi neurony, světlejší modrá (rgb: #0096FF) při najetí myši na konkrétní neuron znázorňuje zvýrazněnou kladnou vazbu mezi neurony a světlejší červená (rgb: #FF0000) znázorňuje zvýrazněnou zápornou vazbu mezi neurony, podobně jako světlejší modrá u kladné vazby. Příklad stylu “Night” viz obr. 2.
- “Day” má světlé pozadí, poněkud tmavší barvy čar znázorňující vazby mezi neurony, kde modrá (rgb: #0000FF) je stejná jak pro základní čáru u kladné vazby, tak pro zvýrazněnou čáru u kladné vazby při najetí myši na neuron. Dále pak tmavá červená (rgb: #C80000) znázorňuje zápornou vazbu mezi neurony a červená (rgb: #FF0000) znázorňuje zvýrazněnou zápornou vazbu. Příklad stylu “Day” viz obr. 3.
- “Print” je verze stylu, kde je vše černobíle, pro případ černobílého tisku. Barva pro čáry zobrazující kladou vazbu je tmavě šedá (rgb: #1D1D1D) a je použita jak pro základní vazbu i pro zvýrazněnou vazbu. Podobně barva pro čáry zobrazující zápornou vazbu je



Obrázek 3: Příklad použití stylu "Day"

jen lehce světlejší šedá (rgb: #3C3C3C) a opět je použita jak pro základní vazbu, tak pro zvýrazněnou vazbu. Aby se kladná a záporná vazba daly od sebe lépe rozeznat, je záporná vazba kreslena čárkovanou čarou. Příklad stylu "Print" viz obr. 4.

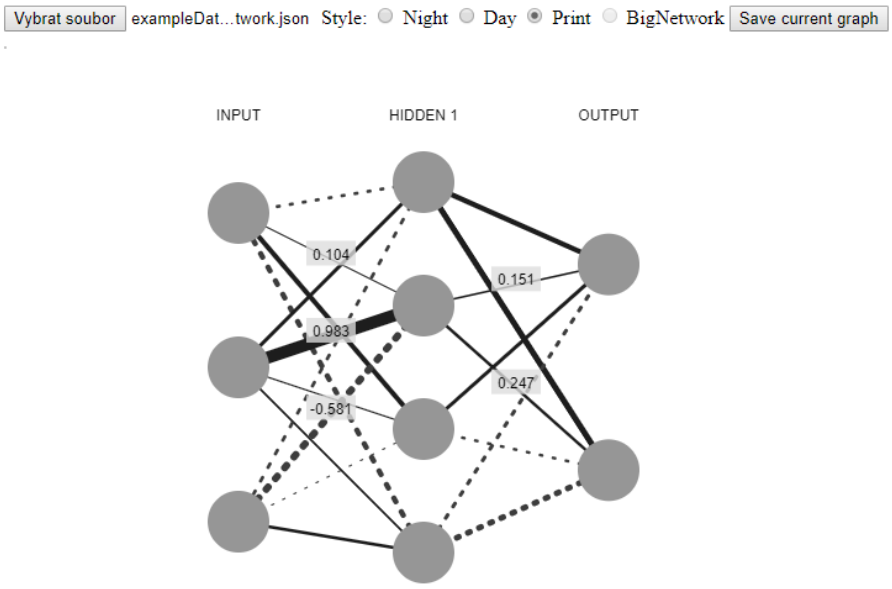
Barevný styl "BigNetwork", jak už název napovídá, nejde použít pro neuronové sítě menší než neuronové sítě s největší vrstvou do 20 neuronů, hlavně z důvodu přehlednosti. Bylo vyzkoušeno, že když je kreslena velká neuronová síť s jedním ze stylů pro malé neuronové sítě viz výše, nelze z nákresu zjistit vůbec nic. Na větší sítě je proto automaticky použit styl "BigNetwork", který je černobílý, má 5x menší neurony, které jsou vertikálně blíže k sobě, ignoruje kladné/záporné vazby, tj. všechny vazby jsou nakresleny jednou barvou o stejné tloušťce. Dále je ignorována kontrola pozice myši ("on hover"), protože při tomto stylu se nezobrazují hodnoty vazeb při najetí myší na neuron, opět z důvodu přehlednosti. U tohoto stylu tedy není nutné neustále překreslovat výsledný obrázek. Příklad tohoto stylu viz obr. 5.

3.2 XY Graphs

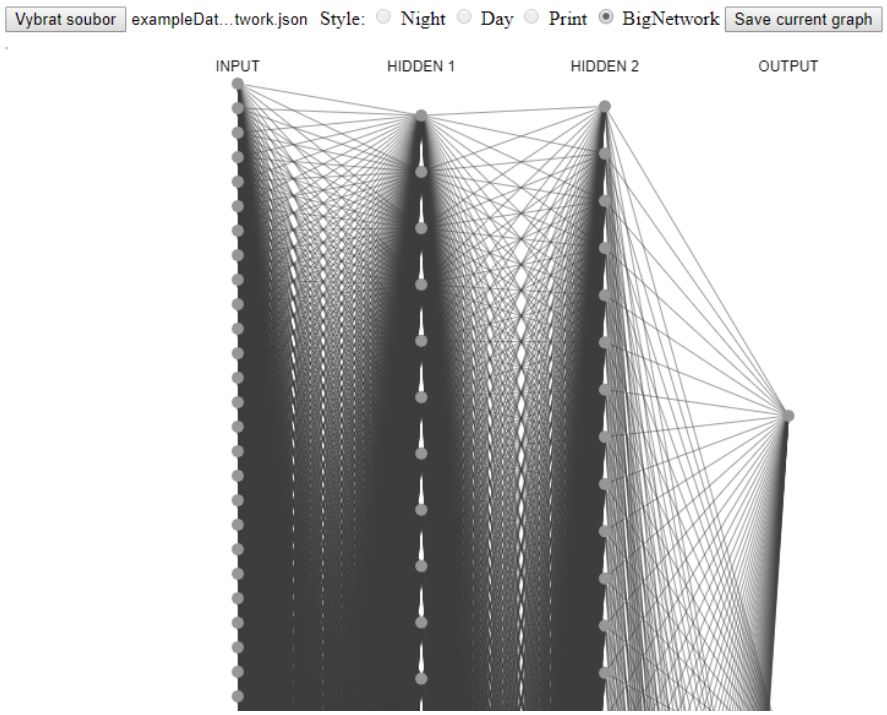
Aplikace zobrazuje XY grafy ze vstupního souboru, použitelné pro zobrazení vývoje ceny nebo přesnosti neuronové sítě při učení. Pro tuto část aplikace byl použit framework Highcharts.

3.2.1 Implementace

Aplikace načte data z uživatelem poskytnutého souboru nebo více souborů s daty, viz kapitola 3.2.2. Data jsou dále převedena na čísla a uložena do 2D pole ve tvaru [číslo epochy, data]. Dále



Obrázek 4: Příklad použití stylu "Print"



Obrázek 5: Příklad použití stylu "BigNetwork"

se vytvoří objekt s daty a dalším nastavením pro framework Highcharts.

```
1  function parseData(temp, filename) {
2      ...
3      for (var i = 0; i < temp.length; i++) {
4          if (isNaN(tempV = parseFloat(temp[i]))) //ignore empty entries, letters
                etc
5              continue;
6          tempVals.push([i / fragmentsPerEpoch, tempV]); //get number (string)
               and parse it into double
7      }
8      ...
9      //first time = create new, then add new data
10     if (chart == undefined) {
11         ShowChart();
12         chart.addSeries(series);
13     }
14     else //if created, just add data
15         chart.addSeries(series);
16     ...
17 }
```

Výpis 5: Zpracování vstupních dat a vytvoření grafu

Pro surová data jsou spočítány statistické údaje, které jsou zobrazeny pod grafem, více viz kapitola 3.2.3.

```
function Variation(values) {
    values = values.map(x => x[1]);
    var avg = values.reduce((a, b) => a + b, 0) / values.length; //average of
        all
    var squareDiffs = values.map(function (value) { //array of (x-x0)^2
        var diff = value - avg;
        var sqrDiff = diff * diff;
        return sqrDiff;
    });
    return avgSquareDiff = squareDiffs.reduce((a, b) => a + b, 0) / (
        squareDiffs.length); //average of squares
}
```

Výpis 6: Výpočet rozptylu

Pokud je přidán další soubor s daty, nevytváří se nový graf, ale pouze se přidají data do již existujícího grafu. Následně jsou pro stejná data spočítány průměry za epochu a ty jsou přidány jako další datová řada do zobrazovaného grafu.

Uživatel má také možnost měnit popisky, viz kapitola 3.2.2. HTML poskytuje vlastnost “contentEditable”, která uživateli umožňuje editovat text v prvku s tímto atributem nastaveným na “true”. Po kliknutí na tlačítko “Save” pod popisky jsou pomocí JavaScriptu přečteny aktuální data z těchto prvků a změny aplikovány na příslušné datové řady a jiné popisky grafu napříč celou záložkou “XY Graphs”.

```
function saveGraphLabels() {
    //series
    var seriesLabels = document.getElementsByClassName("seriesLabel")
    for (var i = 0; i < seriesLabels.length; i++) {
        chart.series[i].update({ name: seriesLabels[i].innerHTML }, false);
    }
    //get current values from html for title and axis'
    graphTitle = labelGraphTitle.textContent;
    graphXAxisLabel = labelXAxis.textContent;
    graphYAxisLabel = labelYAxis.textContent;
    //set new values
    chart.xAxis[0].setTitle({ text: graphXAxisLabel });
    chart.yAxis[0].setTitle({ text: graphYAxisLabel });
    chart.setTitle({ text: graphTitle });
    ...
}
```

Výpis 7: Uložení popisků zadaných uživatelem

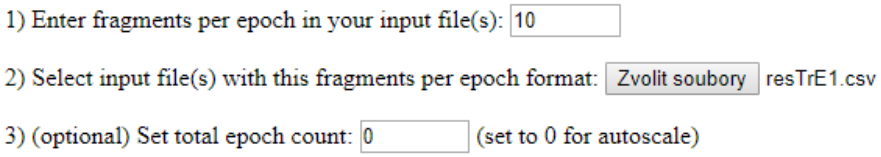
3.2.2 Vstupní data

Aplikace na vstupu očekává soubor s daty typu double, jedno číslo na řádku (preferovaný formát .csv). Tato data budou zobrazena na ose Y v grafu, osa X je dána pořadím čísla v souboru. Příklad vstupních dat viz výpis č. 8.

```
8.906043763535317
3.4196297263061983
3.0229553412595593
2.9887189110315733
2.976132697678099
2.863084248954071
2.8680988867049004
2.6848446986104197
```

Výpis 8: Vstupní data pro XY grafy data

Uživatel při nahrávání souboru zadává, kolik po sobě jdoucích hodnot patří k jednomu fragmentu učení neuronové sítě, protože ze vstupních dat toto nelze určit. Aplikace do jednoho grafu zobrazuje surová vstupní data tak, jak jsou ve vstupním souboru, a data zprůměrovaná



Obrázek 6: Zadání vstupních dat

za 1 epochu v sobě odpovídajících barvách – světlejší barvou jsou surová data a k nim odpovídající tmavší barvou jejich průměry. Do aplikace lze nahrát až 5 různých vstupních souborů, a to i s různým počtem fragmentů na epochu, jen je nutno vždy před nahráním dalšího souboru zvolit počet fragmentů na periodu pro daný soubor. S více než 5 vstupními soubory se graf stává poněkud nepřehledným. Aplikace ignoruje řádky, které neobsahují čísla, a tyto řádky ani nepočítá do fragmentů pro danou epochu, ve které se takový řádek vyskytuje.

3.2.3 Ovládání

Uživatel nejprve zadá počet fragmentů na epochu v nahrávaném vstupním souboru, poté nahraje soubor. Aplikace následně vykreslí graf z daty ze souboru. Uživatel má možnost nastavit maximální počet zobrazovaných epoch, aplikace pak "ustřihne" graf za touto hodnotou. Pokud uživatel nezadá tuto hodnotu je počet zobrazovaných epoch počítán ze vstupního souboru.

Vykreslený graf má možnost přiblížení ve směru osy x tak, že uživatel myší označí data, která chce zvětšit, a to pomocí "drž a táhni" povelu myší, viz šedá plocha na obr. 7. Po najetí myší na bod zobrazené datové řady jsou zobrazeny další informace o bodu – číslo epochy, hodnota bodu a jeho popisek. Pokud více bodů má stejnou souřadnici na ose x (sub-epochu), je zobrazeno jen jedno okno s informacemi o všech bodech splňujících tuto podmínku. Graf také obsahuje menu s možností stažení právě zobrazeného grafu, a to ve formátech PNG, JPG, PDF a SVG, případně je možné graf vytisknout.

Vedle zobrazeného grafu je možnost editovat jednotlivé prvky grafu – název, popisky os a popisky jednotlivých datových řad. Změny zde se projeví napříč celou záložkou "XY Graphs".

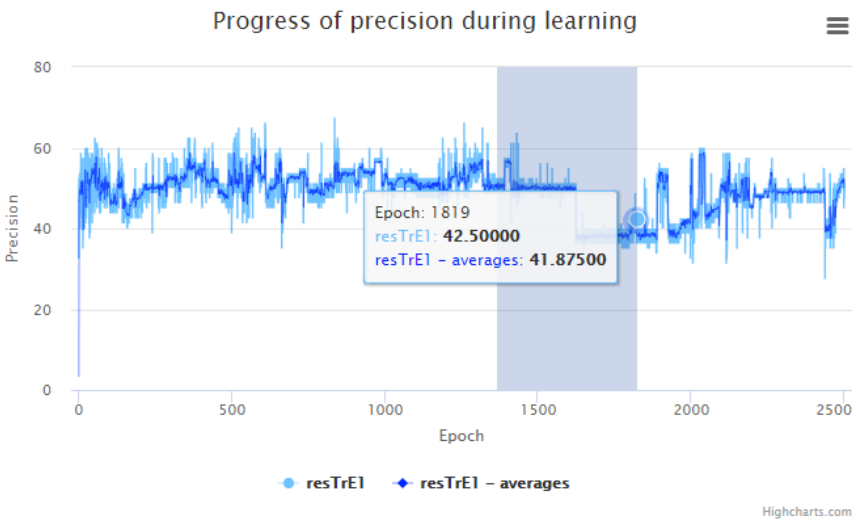
Pod grafem jsou zobrazeny statistické informace k nahraným datovým sadám obsahující

- rozptyl podle vzorce

$$Var(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

- směrodatnou odchylku podle vzorce

$$\sigma = \sqrt{Var(X)} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$



Obrázek 7: Ukázkový graf s oblastí vybranou pro zoom

Edit graph labels here:

Graph title: Progress of precision during learning

X-axis title: Epoch

Y-axis title: Precision

Series title(s):

resTrE1

resTrE1 - averages

Save

Obrázek 8: Oblast pro editaci popisků

Statistic info:

resTrE1:

Variation: 27.538127159998115

Standard deviation: 5.247678263765616

Obrázek 9: Zobrazení statistických dat

3.3 Confusion Matrix

Aplikace zobrazí Matici záměn pro data z neuronové sítě. Dále pak spočítá další informace (četnosti True positive, False positive, False negative, True negative, jejich relativní četnosti, Precision, Recall, F1 score, Specificity, G-Mean, Matthews correlation coefficient – MCC, vzorce viz kapitola 2.3.3) a zobrazí je v tabulce. Dále pak porovná oba vstupní soubory a označí chybné řádky. Pro zobrazení Matice záměn byl použit framework Highcharts.

3.3.1 Implementace

Aplikace na vstupu očekává 2 vstupní soubory:

- Soubor s opravdovými výsledky
- Soubor s výsledky klasifikace neuronovou sítí

Tyto soubory aplikace načte, porovná jednotlivé řádky a výsledky tohoto porovnání uloží do pole. Následně jsou data z tohoto pole převedena do formátu pro framework Highcharts.

```
1  function doTheMagicCounting() {
2      ...
3      dataTrueLabels = [...new Set(dataTrue)].sort(); //get unique categories
           from true class and sort them
4      dataPredictedLabels = [...new Set(dataPredicted)].sort(); //get unique
           categories from prediction class and sort them
5
6      data = make2DArray(dataTrueLabels.length, dataPredictedLabels.length, 0);
           // create empty 2d array to store comparison results
7
8      for (var i = 0; i < dataPredicted.length && dataTrue.length; i++) { //check
            all lines
9          var indexTrue = dataTrueLabels.indexOf(dataTrue[i]); //select col with
               checked true class
10         var indexPredicted = dataPredictedLabels.indexOf(dataPredicted[i]); //
               select row with prediction class
11         data[indexPredicted][indexTrue]++; //yay we got match so ++
12     }
13     ...
14 }
```

Výpis 9: Zpracování vstupních dat pro Matici záměn

Output analysis

Name	TP	FP	FN	TN	TPrate	FPrate	FNrate	PRE	REC	F1	SPE	G-Mean	MCC
call_test	1869	4	7	1597	0.996	0.002	0.004	0.998	0.996	0.997	0.998	0.997	0.994
opt_scan	232	3	3	3239	0.987	0.001	0.001	0.987	0.987	0.987	0.999	0.987	0.986
opt_test	30	14	0	3433	1.000	0.004	0.000	0.682	1.000	0.811	0.996	0.826	0.824
reg&call	202	13	10	3252	0.953	0.004	0.003	0.940	0.953	0.946	0.996	0.946	0.943
reg_attempt	50	64	0	3363	1.000	0.019	0.000	0.439	1.000	0.610	0.981	0.662	0.656
reg_test	345	7	90	3035	0.793	0.002	0.029	0.980	0.793	0.877	0.998	0.882	0.867
reg_test_high	555	5	5	2912	0.991	0.002	0.002	0.991	0.991	0.991	0.998	0.991	0.989
ukwSIP/noSIP	75	9	4	3389	0.949	0.003	0.001	0.893	0.949	0.920	0.997	0.921	0.919
	3358	119	119	24220	0.959	0.005	0.005	0.864	0.959	0.892	0.995	0.901	0.897

Total: 3477

Ok: 3358

Bad: 119

Correct classification: 96.5775 %

Error rate: 3.4225 %

Obrázek 10: Příklad zobrazené tabulky s dalšími informacemi

Pod tuto tabulku jsou vypsány krátké informace o zobrazované sadě: Počet správných, chybných řádků, celkový počet řádků, úspěšnost a chybovost. Příklad viz obr. 10.

3.3.2 Vstupní data

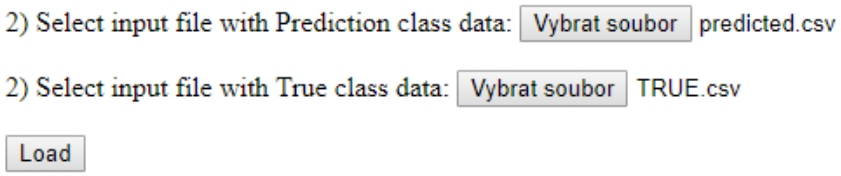
Aplikace očekává od uživatele 2 soubory: jeden s reálnými výsledky a jeden výsledky klasifikace neuronovou sítí. Formát těchto souborů je textový, jeden řádek je brán jako jeden řetězec, což odpovídá jednomu výsledku.

```
1 reg_test
2 reg_test_high
3 reg_test_high
4 reg_test_high
5 reg_test_high
6 reg_test_high
7 reg_test_high
```

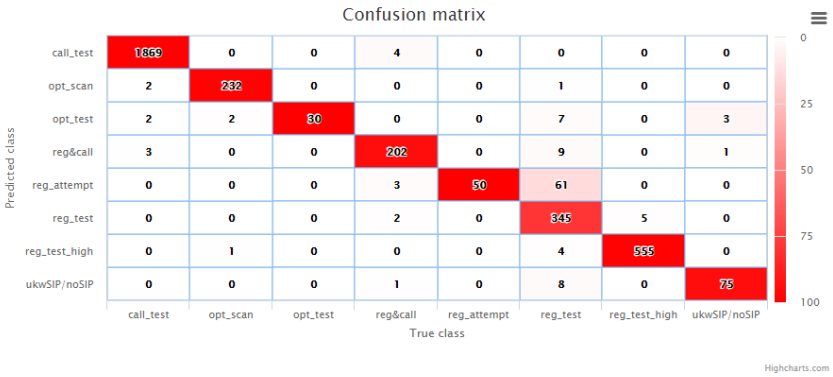
Výpis 10: Příklad vstupního souboru pro Matici záměn

3.3.3 Ovládání

Ovládání této části je velmi přímočaré, uživatel si musí dát pozor na správné zadání vstupních souborů, pokud je zadá do špatného vstupního pole, budou výsledky špatně popsány. Aplikace nemá jak rozlišit data pro jednotlivé třídy, toto náleží uživateli. Po kliknutí na tlačítko “Load” je zahájen výpočet.



Obrázek 11: Formulář pro vstupní data záložky Confusion Matrix



Obrázek 12: Matice záměn

Vykreslená Matice záměn má možnost uložení aktuální Matice v několika formátech (PNG, JPG, PDF, SVG) nebo možnost tisku Matice záměn. Příklad Matice záměn viz obr. 12.

3.3.4 Zobrazované styly

Uživatel si může vybrat ze dvou stylů:

- Barevný styl, kde jednotlivé buňky v Matici záměn jsou podbarveny podle procentuální četnosti výskytu výsledků v jednotlivých sloupcích matice. Záhlaví dalších tabulek je také podbarveno a jednotlivé řádky barevně odlišeny světlým podbarvením.
- Černobílý, kde barvy nejsou použity ani u Matice záměn, ani u dalších tabulek. Vhodné k černobílému tisku, nicméně horší orientace v tabulkách i Matici záměn – buňky nejsou podbarveny.

3.4 Output Class

Aplikace zobrazuje nespojový XY bodový graf s daty podle výběru uživatele.

3.4.1 Implementace

Po načtení vstupního souboru jsou data rozdělena podle příslušnosti do tříd podle posledního sloupce, viz kapitola 3.4.2. Následně jsou numerická data převedena na čísla a vše je uloženo

do pole 2D polí tak, že na indexu *data[i]* je uloženo 2D pole s daty i-té unikátní třídy, řazeno dle abecedy. Následně jsou vytvořeny rozbalovací nabídky pro uživatele k výběru požadovaných sloupců dat.

Ve výchozím stavu bez zásahu uživatele se kreslí graf s daty z prvního sloupce jak na ose x, tak na ose y. Více k ovládání viz kapitola 3.4.3.

Při implementaci této části aplikace byly zaznamenány jisté problémy s výkonem aplikace a to hlavně z důvodu překreslování grafu při přidání každé jednotlivé datové sady (třídy) a při změně názvů os. Graf se nyní překresluje až po změně všech datových sad a názvy os se mění jen jednou za změnu dat, ne při každé změně datové sady. Dále bylo vylepšeno zpracování vstupních dat tak, aby při následném výběru požadovaného sloupce aplikace nemusela procházet všechna data. Díky těmto optimalizacím bylo dosaženo zlepšení výkonu 2x-4x. Se vstupní datovou sadou obsahující 320 řádků a 11 sloupců trvalo před optimalizací načtení a zobrazení dat 4 sekundy, výběr jiných dat a jejich zobrazení pak 4,2 sekundy. Po optimalizaci trvá načtení < 1 sekundu a výběr jiných dat a jejich zobrazení < 1.5 sekundu. Měřeno vývojářskými nástroji v prohlížeči Chrome 73 (64bit), stroj s procesorem Intel i5-3337U, 8 GB RAM.

3.4.2 Vstupní data

Aplikace na vstupu očekává soubor ve formátu .csv s daty k zobrazení. Na prvním řádku souboru je záhlaví s popisem jednotlivých sloupců tabulky, viz výpis 11.

```
1 delta,CC,RC,IC,AC,BC,CC,OC,SC,rate,class
2 0,1,0,0,0,0,0,3,0,3,opt_test
3 3,1,0,0,0,0,0,2,0,2,opt_test
4 1,6,0,0,0,0,0,2,0,0.67,opt_test
5 1,2,0,0,0,0,0,1,0,0.5,opt_test
```

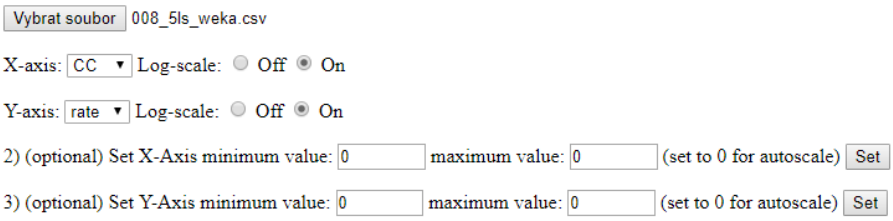
Výpis 11: Příklad vstupního souboru pro Output Class

Jednotlivé hodnoty jsou odděleny čárkou, očekávaný formát jsou reálná čísla, mimo posledního sloupce, ve kterém je uveden název třídy, ke kterému data náleží.

3.4.3 Ovládání

Po nahrání souboru je uživateli zobrazen ovládací panel s nastavením jednotlivých os zobrazovaného grafu. Uživatel má možnost zvolit si, jaká data budou zobrazena na jednotlivých osách grafu, zda bude mít osa logaritmické měřítko a rozsah jednotlivých os. Při zadání nuly pro nastavení rozsahu os je použito automatické měřítko na základě minimální a hlavně maximální hodnoty zobrazovaných dat. Uživatel má také možnost myší označit oblast, kterou chce v grafu přiblížit (funkce "zoom"). Graf má možnost uložení aktuálního grafu v několika formátech (PNG,

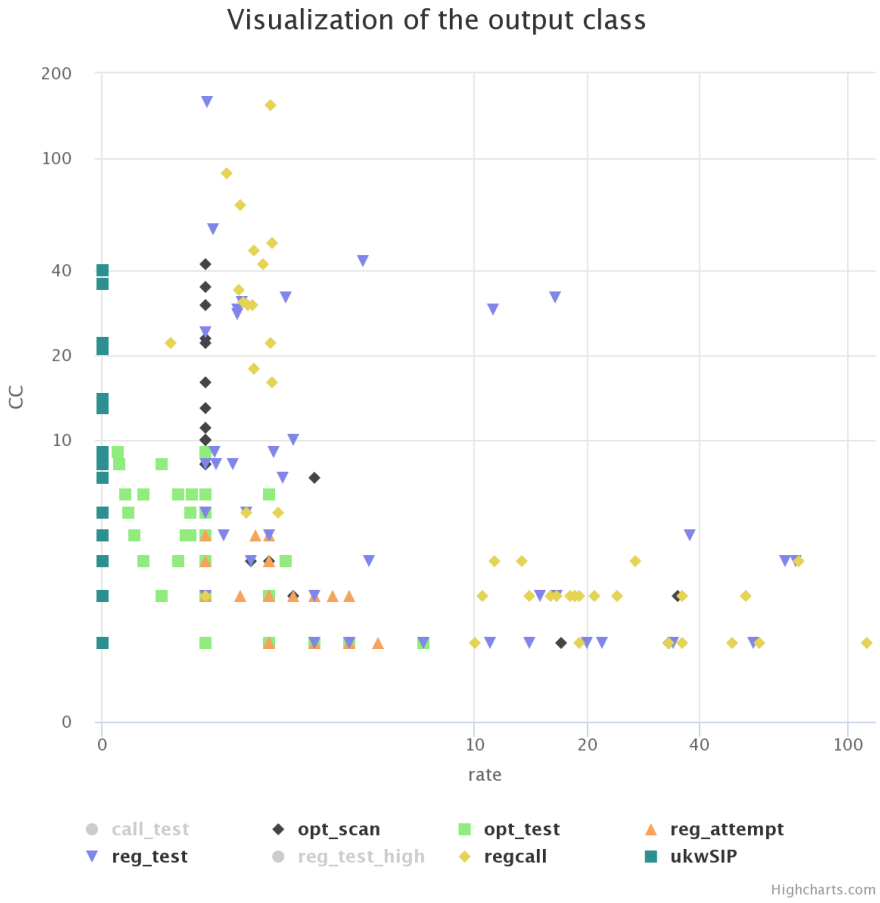
JPG, PDF, SVG) nebo možnost tisku. Dále může uživatel skrýt nebo zobrazit jednotlivé datové sady kliknutím na legendu pod grafem.



Obrázek 13: Ovládání části Output Class

3.4.4 Výstup

Výstupem této části aplikace je "rozptylový"graf, tj. nespojový XY bodový graf zobrazující jakoukoliv dvojici uživatelem zvolených dat v ploše. Jednotlivé datové sady (třídy) jsou barevně odlišeny. Příklad výstupního grafu viz obr. 14.



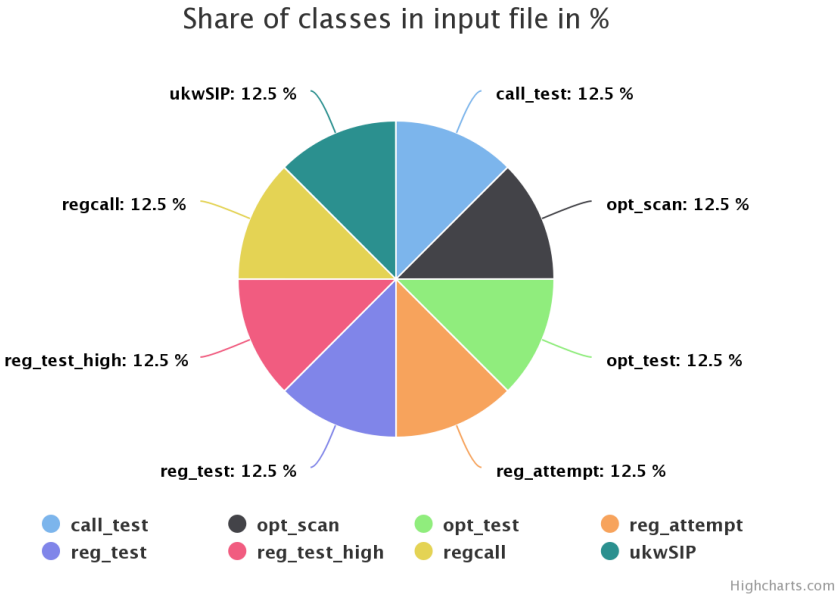
Obrázek 14: Výstupní graf části Output Class

3.5 Input Class

Tato část aplikace se dá rozdělit na 2 podčásti. První podčást je specifická pro tuto část aplikace, druhá je funkčně velmi podobná části aplikace Output Class. Jediný rozdíl u druhé podčásti jsou barvy u datových sad (tříd) u nespojového XY bodového grafu.

3.5.1 Implementace

Po načtení vstupního souboru jsou data rozdělena stejně jako u části Output Class, viz kapitola 3.4.1. Navíc je vypočítáno procentuální zastoupení jednotlivých tříd ve vstupním souboru a tato data jsou zobrazena v koláčovém grafu.



Obrázek 15: Koláčový graf s procentuálním zastoupením tříd ve vstupním souboru

Také jsou vypočítány statistiky nejprve celkové pro všechny třídy dohromady, následně i pro jednotlivé třídy, a to po sloupcích, tj. jednotlivých položek u tříd. Konkrétně je vypočítána minimální a maximální hodnota, suma, průměr, rozptyl a směrodatná odchylka. Rozptyl a směrodatná odchylka podle vzorce viz kapitola 3.2.3. Vše je zobrazeno do přehledné tabulky, kde na řádcích jsou jednotlivé třídy a jejich položky, ve sloupcích pak vypočítaná data.

3.5.2 Vstupní data

Vstupní soubor má přesně stejný formát jako u části aplikace Output Class, viz kapitola 3.4.2, ukázka viz výpis 11.

Statistics

class	Name	Total	Min	Max	Avg	Var	StdDev
Total	CC	3924.000	1.000	556.000	12.262	1820.181	42.664
	RC	468003.000	0.000	246720.000	1462.509	201004636.644	14177.610
	IC	2367.000	0.000	277.000	7.397	767.852	27.710
	AC	1164.000	0.000	347.000	3.638	624.900	24.998
	BC	20.000	0.000	10.000	0.063	0.521	0.722
	CC	624.000	0.000	88.000	1.950	61.473	7.840
	OC	664.000	0.000	70.000	2.075	39.082	6.252
	SC	4.000	0.000	4.000	0.013	0.050	0.223
	rate	212176.690	0.000	123361.000	663.052	49031024.387	7002.216
	CC	1460.000	1.000	556.000	36.500	10997.700	104.870

Obrázek 16: Zobrazení tabulky se statistickými údaji

3.5.3 Ovládání

Ovládání první z podčástí je velmi jednoduché. Uživatel nahraje vstupní soubor, což spustí výpočet. Uživatel má také možnost myší kliknout na oblast v koláčovém grafu a tím ji zvýraznit. Graf má možnost uložení aktuálního grafu v několika formátech (PNG, JPG, PDF, SVG) nebo možnost tisku. Dále může uživatel skrýt nebo zobrazit jednotlivé datové sady kliknutím na legendu pod grafem. Ovládání druhé podčásti je shodné s částí aplikace Output Class, viz kapitola 3.4.3, příklad výstupního grafu viz obr. 14.

3.6 Binary Input Class

Tato část aplikace zobrazuje tabulku s obarvenými buňkami podle dat ze vstupního souboru.

3.6.1 Implementace

Po načtení vstupního souboru je vytvořeno 2D pole s daty v paměti programu. Následně je vytvořena tabulka s buňkami obarvenými podle hodnot v poli. Pokud se liší po sobě jdoucí řádky v buňce třída, je vložen prázdný řádek s názvem této třídy jako titulek pro následující řádky. K buňkám je přidán popisek s řádkem, sloupcem a třídou dané buňky.

3.6.2 Vstupní data

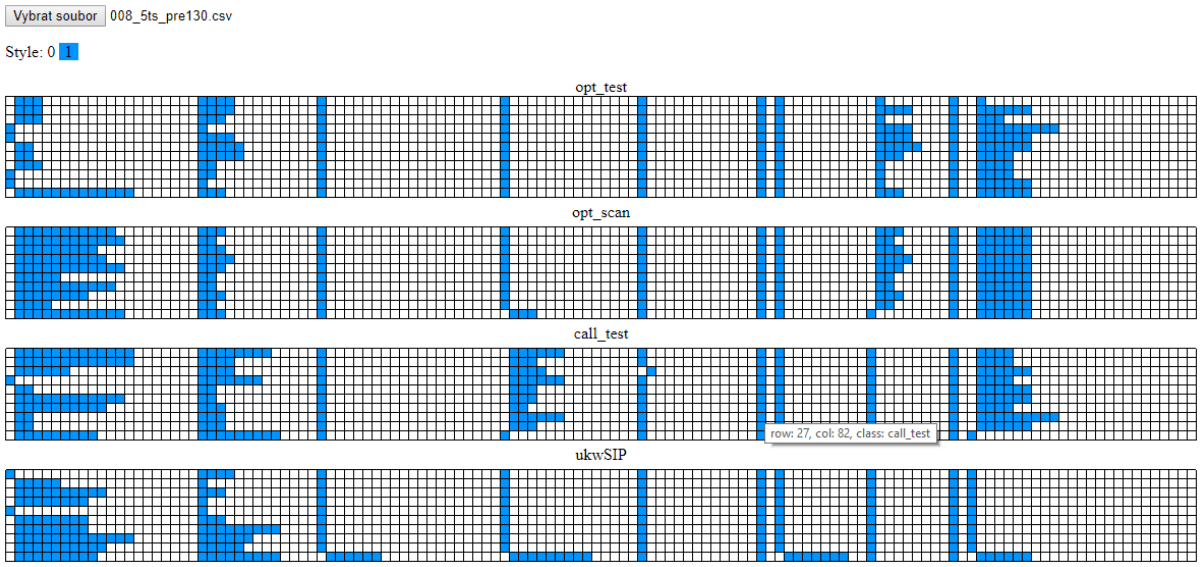
Aplikace očekává soubor typu .csv s binárními daty, tj. hodnotami 0 nebo 1. V poslední buňce řádku je uveden název třídy, ke které daný řádek patří. Aplikace očekává, že vstupní soubor je setříděn podle tříd. Oddělovačem je středník.

```
1;1;1;1;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;opt_test
1;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;opt_test
1;1;1;1;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;opt_scan
1;1;1;1;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;opt_scan
```

Výpis 12: Příklad vstupního souboru pro Binary Input Class

3.6.3 Ovládání

Uživatel nahraje vstupní soubor. Ihned po zadání tohoto souboru je proveden výpočet, viz kapitola 3.6.1. Uživateli je zobrazena tabulka s daty, kde hodnotu 0 představuje neobarvená buňka, hodnotu 1 obarvená buňka. Po najetí na konkrétní buňku je uživateli zobrazena informace o řádku a sloupci ze vstupního souboru a třídě, kam daná buňka patří.



Obrázek 17: Zobrazení výstupní tabulky u Binary Input Class

4 Dokumentace

K programu vytvořeného pro tuto práci byla vytvořena dokumentace, která je přílohou této práce.

5 Závěr

Hlavním úkolem této práce bylo vytvoření vizualizačního nástroje pro zobrazení vnitřní struktury umělé neuronové sítě, grafů vývoje učení a analýza vstupních a výstupních dat pomocí technologií pracujících ve webovém prostředí. Proto byla zvolena forma webových stránek (technologie HTML) s dynamickými prvky ovládanými pomocí jazyka JavaScript.

Vizualizační nástroj byl od počátku vyvíjen nad reálnými daty z umělé neuronové sítě projektu běžícího na Katedře telekomunikační techniky.

Při vývoji aplikace jsem se setkal s problémy tykajícími se výkonu aplikace hlavně v první části aplikace při vykreslování velkých sítí. Výsledný graf vnitřní struktury sítě se stal také poněkud nepřehledným, byla-li na vstupu zadána příliš velká síť. Proto byl implementován styl „BigNetwork“, který řeší tyto problémy. Další problém s výkonem byl v částech čtyři a pět při vykreslování nespojového XY bodového grafu. Graf se původně překresloval při přidání nebo odebrání každé datové řady (vstupní třídy) a to hned 2x – přidání datové řady a nastavení popisů os, při testování tedy i šestnáctkrát za změnu zobrazovaných dat (při osmi třídách ve vstupním souboru). Nyní se graf překresluje jen jednou po přidání všech datových sad. Vývoj zbylých částí aplikace proběhl bez větších překážek.

Protože nástroj byl vyvíjen pro projekt běžící na katedře, předpokládám použití tohoto vizualizačního nástroje minimálně v tomto projektu, případnou integraci celého nástroje do webového rozhraní pro komplexní práci s neuronovou sítí.

Literatura

[1] AIGER, Wolfgang, Silvia MIKSCH, Heidrun SCHUMANN a Christian TOMINSKI. *Visualization of time-oriented data*. London: Springer, c2011, s. 3-4. Human-computer interaction series. ISBN 978-0-85729-078-6.

[2] MCCORMICK, Bruce H., Thomas A. DEFANTI a Maxine D. BROWN. Visualization in Scientific Computing. *Computer Graphics*. 1987, 21(6), 3. Dostupné také z: http://www.sci.utah.edu/vrc2005/McCormick-1987-VSC.pdf

[3] ROBERT, Spence. *Information Visualization: Design for Interaction*. 2. vydání. Harlow: Pearson/Prentice Hall, 2007. ISBN 978-0-13-206550-4.

[4] WARD, Matthew, Georges G. GRINSTEIN a Daniel KEIM. *Interactive Data Visualization: Foundations, Techniques, and Applications*. Natick, Mass.: A K Peters, c2010. ISBN 978-1-56881-473-5.

[5] WONG, Pak Chung a R. Daniel BERGERON. 30 years of multidimensional multivariate visualization. NIELSON, Gregory M., H. HAGEN a Heinrich MÜLLER, ed. *Scientific visualization: overviews, methodologies, and techniques*. Los Alamitos, Calif.: IEEE Computer Society, c1997, s. 3-33. ISBN 0-8186-7777-5.

[6] Tisková zpráva: *Netscape and Sun announce JavaScript* [online]. 4.12.2005 [cit. 2019-04-04]. Dostupné z: https://web.archive.org/web/20070916144913/http://wp.netscape.com/newsref/pr/newsrelease67.html

[7] Standard ECMA-262 archive. *ECMA International* [online]. [cit. 2019-04-04]. Dostupné z: https://www.ecma-international.org/publications/standards/Ecma-262-arch.htm

[8] P5.js API Reference [online]. [cit. 2019-04-04]. Dostupné z: https://p5js.org/reference/

[9] Highcharts API Reference [online]. [cit. 2019-04-04]. Dostupné z: https://api.highcharts.com/highcharts/

[10] FAWCETT, Tom. An introduction to ROC analysis. *Pattern Recognition Letters* [online]. 2006, 27(8), 861-874 [cit. 2019-04-18]. DOI: 10.1016/j.patrec.2005.10.010. ISSN 01678655. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S016786550500303X

[11] ISO 19157:2013. *Geographic Information – Data Quality*. Geneva: International Organization for Standardization, 2013.

[12] F.J, Ariza-Lopez, Rodri guez-Avi J a Alba-Ferna ndez M. V. Complete Control of an Observed Confusion Matrix. In: IGARSS 2018–2018 *IEEE International Geoscience and Remote*

Sensing Symposium [online]. Valencia, Spain: IEEE, 2018, 2018, s. 1222-1225 [cit. 2019-04-18]. DOI: 10.1109/IGARSS.2018.8517540. ISBN 978-1-5386-7150-4. ISSN 2153-7003. Dostupné z: https://ieeexplore.ieee.org/document/8517540/

[13] MCCULLOCH, Warren S. a Walter PITTS. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* [online]. 1943, 5(4), 115-133 [cit. 2019-04-12]. DOI: 10.1007/BF02478259. ISSN 0007-4985. Dostupné z: http://link.springer.com/10.1007/BF02478259

[14] Neural network. *DeepAI* [online]. [cit. 2019-04-18]. Dostupné z: https://deepai.org/machine-learning-glossary-and-terms/neural-network

A Přílohy na přiloženém CD

Přílohy této bakalářské práce jsou uloženy na přiloženém CD

(a) Zdrojové kódy aplikace

(b) Ukázkové vstupní soubory

(c) Programátorská dokumentace ve formátu PDF